DFAL
User's Guide

By

Hongyi Li, Ping Tang and Ling Ding
Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing, 100101, China
Datun Rd., Chaoyang District Beijing Po.Box. 9718, 100101.China.

1. DFAL Summary

1.1. DFAL format supported

The DFAL interface is extended from the original GDAL, HDF5 and HDF4 libraries, and supports all data types supported by L1B, HDF5, HDF4 and GDAL. It is shown in Table 1.

Table 1 Data types supported

Data type	Reading	Writing
L1B	OK	NO
HDF5	OK	OK
HDF4	OK	OK
GDAL	OK	OK

1.2. DFAL software required

Table 2 Software required

No	Property	Name	Version
1	Systems software	Ubuntu 64 bit	12.04 LTS Server
2	Editing software	CodeBlocks	10.05
3	Compiling software	g++	4.6.3
4	Development software	GDAL 64 bit	1.9.2
5	Development software	HDF5	1.8.11
6	Development software	HDF4	4.2.9
7	Development software	proj	4.8

2. DFAL configure

2.1. DFAL generated

If you don't want to compile, you can skip this step directly.
In GDAL, the library files of GDAL, HDF5 and HDF4 are downloaded from the official website, among which HDF5 and HDF4 can download 64-bit binary libraries directly, but GDAL libraries are source code and need to be compiled by themselves.
The compilation command of GDAL is as follows:
~$:. / configure
$: make
~$: make install
Following installation, include and lib folders will be generated in usr\ local directory. The two folders will be placed in the new GDAL folder, and the downloaded HDF5 and HDF4 library files will be placed in the generation directory of DFAL. Installation is over.

2.2. Adding DFAL supported to the programs

1. Place the GDAL library, HDF5 library and DFAL Library in the generated directory of the new project, as shown in Figure 1:

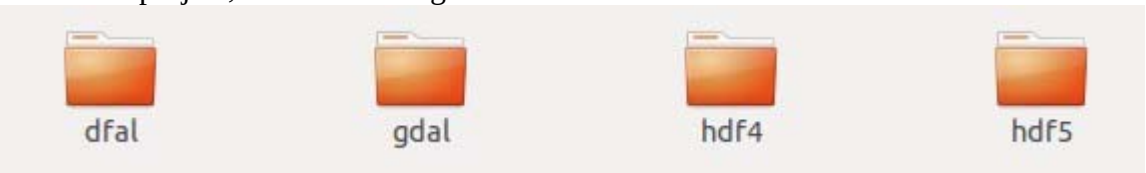


Figure 1 The generated directory of the new project

2. Click right mouse button on the project name and pop up the dialog box as shown in Figure 2.

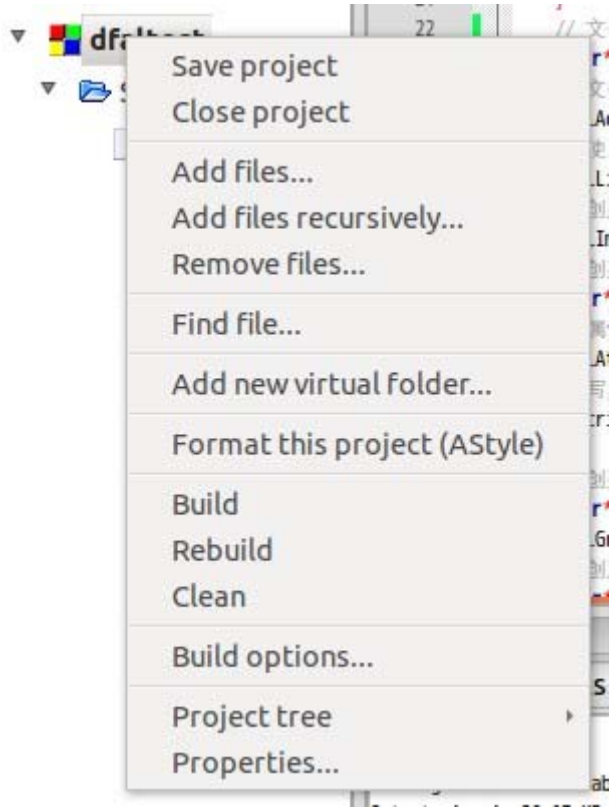
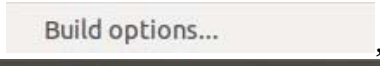


Figure 2 The dialog box

3.Select 

, and pop up the dialog box as shown in Figure 3.

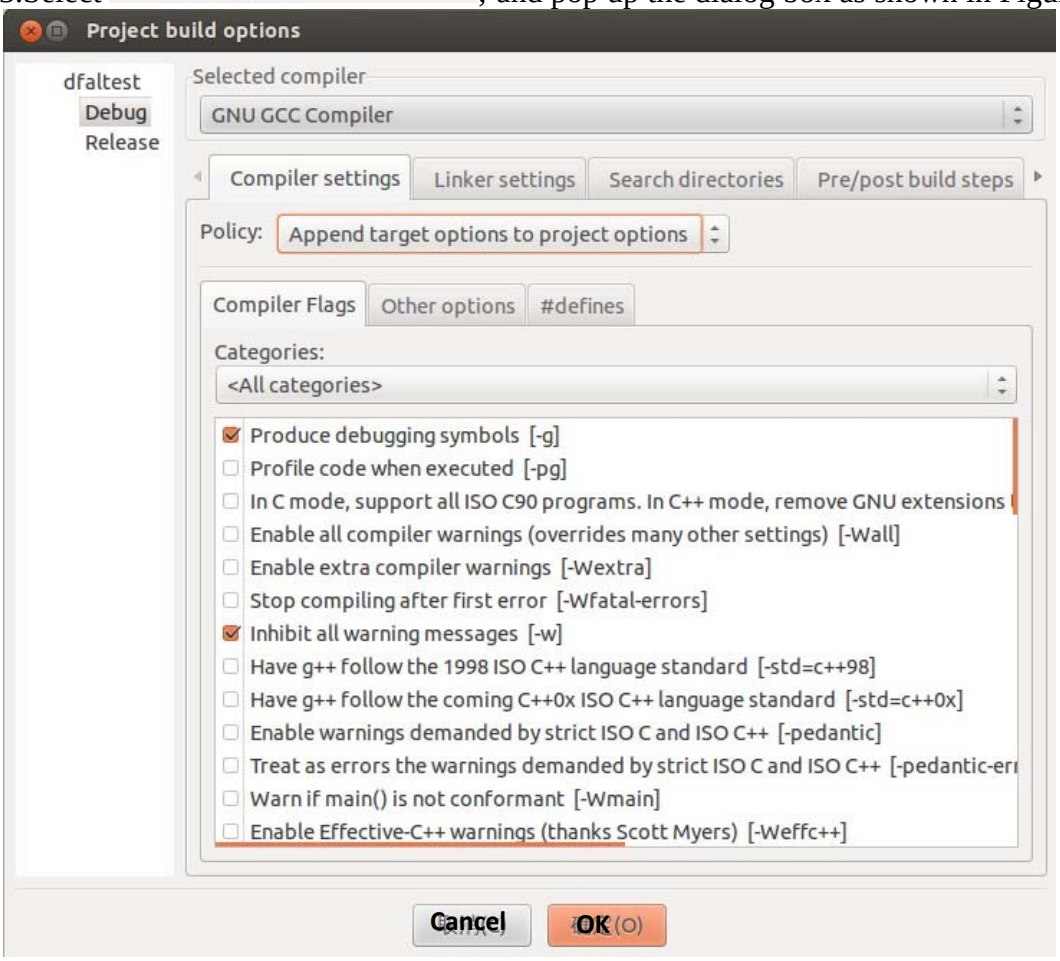
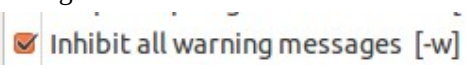
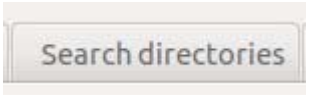


Figure 3. The dialog box

4.Select 

, and then check 

.The function of this option is not to let the compiler treat warnings as errors. If you think your program will not have warnings, you can uncheck them.

5. Select 

, The interface appears as shown in Figure 4.

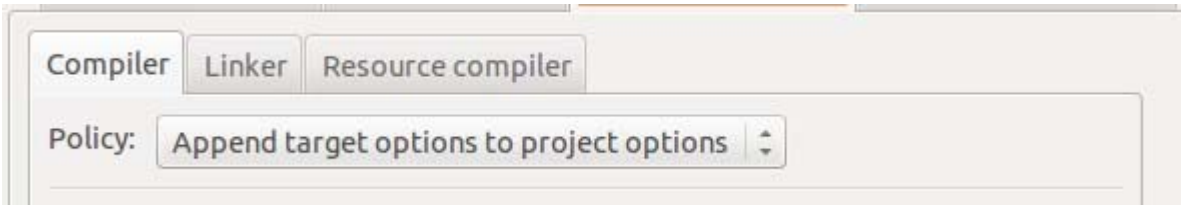


Figure 4 The interface

6. Select 

, add three include directories to the interface, as shown in Figure 5.

Figure 5 Three include directories

7.Select 

, add three lib directories to the interface, as shown in Figure 6.

Figure 6 Three lib directories

8. Select 

, add four libraries to the interface, as shown in Figure 7. Note: Remove “lib” before library name and “so” after library name.

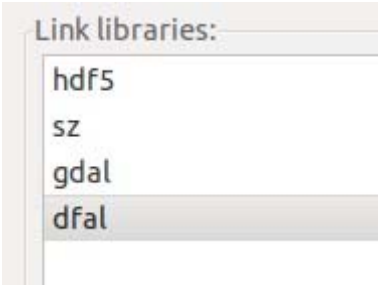


Figure 7 Link libraries

9.Click 

, Configuration completed.

10. The program codes include header files `#include "util external dfal.h"` .

3.Attributes in DFAL

3.1 Data Types

```
// Data type of virtual interface
typedef enum {
    DFAL_DT_Unknown = 0,
    DFAL_DT_Int8 = 1,
    DFAL_DT_UInt8 = 2,
    DFAL_DT_UInt16 = 3,
    DFAL_DT_Int16 = 4,
    DFAL_DT_UInt32 = 5,
    DFAL_DT_Int32 = 6,
    DFAL_DT_Float32 = 7,
    DFAL_DT_Float64 = 8,
```

```
    DFAL_DT_CHAR = 9
} DFALDataType;
```

3.2 File access mode

```
typedef enum {
    // Only reading
        DFAL_AC_ReadOnly = 0,
        // reading and writing
        DFAL_AC_ReadWrite = 1,
        // Create nonexistent
        DFAL_AC_Create = 2,
        // Create and override the original
        DFAL_AC_TRUNC = 3
} DFALAccess;
```

3.3 Library dependency

```
typedef enum {
        // Dependent on GDAL libraries
        DFAL_LT_GDAL = 0,
       // Dependent on HDF4 libraries
         DFAL_LT_HDF4 =1，
        // Dependent on HDF5 libraries
        DFAL_LT_HDF5 = 2,
        DFAL_LT_L1B = 3
} DFALLibraryType;
```

3.4Error formats

```
// Error Types of virtual Interface in DFAL
typedef enum {
        DFAL_Err_None = 0,
        DFAL_Err_Debug = 1,
        DFAL_Err_Warning = 2,
        DFAL_Err_Failure = 3,
        DFAL_Err_Fatal = 4,
 // An error caused by calling a virtual class indicates that the function is not overloaded by the corresponding subclass, that is, the subclass does not support this method.
        DFAL_Err_Vitural = 5
} DFALErr;
```

4. Interface classes and functions of DFAL

4.1. Class structure of DFAL

There are five basic classes: image file class DFALImage, data group class DFALGroup, datasets class DFALDataSet, band class DFALBand, attribute class DFALAttribute.HDF5 package class implements four of them, respectively is image file class HDF5DFALImage, group class HDF5DFALGroup, data set class HDF5DFALDataSet, attribute class HDF5DFALAttribute for HDF5;HDF4 package class implements three of them: image file class HDF4DFALImage, dataset class HDF4DFALDataSet,attribute class HDF4DFALAttribute for HDF4;The GDAL packaging class implements three of them: GDAL image file class GDALDFALImage, GDAL dataset class GDALDFALDataSet, GDAL band class GDALDFALBand; L1B packaging class implements three of them, L1B image file class L1BDFALImage, L1B package class implements three of them, namely image file class L1BDFALImage, dataset class L1BDFALDataSet, band class L1BDFALBand for L1B.

4.2. External interface architecture of DFAL

DFAL can only get DFALImage class through two functions in util_external_dfal.h header file, and then get DFALGroup, DFALDataSet, DFALBand, DFALAttribute according to the class structure level according to section 4.1, and read and write files/groups/datasets/bands/attributes.

The function to open an existing file is as follows:

DFALImage* OpenImageFile(char* strFileName,DFALAccess accessMode,DFALLibraryType libraryType);

The function to create a new file is as follows:

DFALImage* CreateImageFile(char* strFileName,DFALAccess accessMode,DFALLibraryType libraryType,char* strFileFormat);

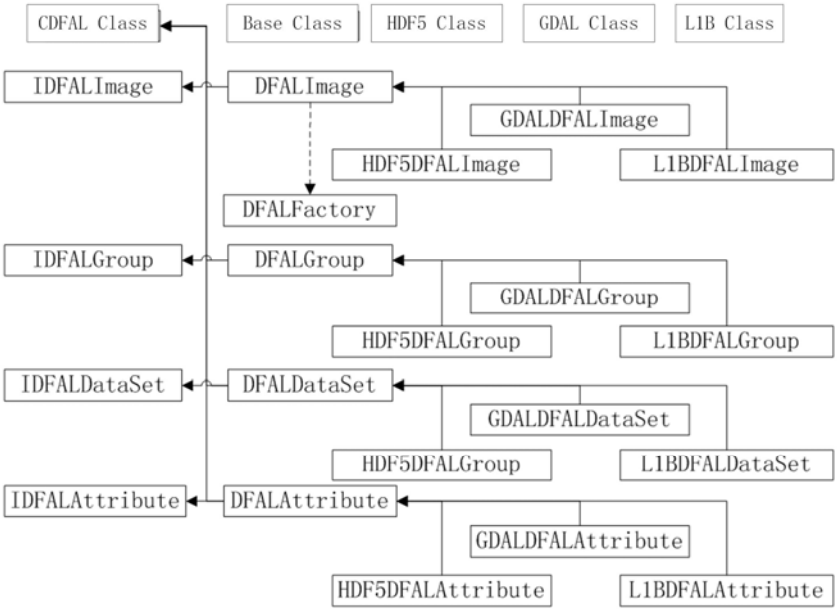


Figure 8 Class architecture of DFAL

4.3. DFALImage class interface

```
class DFALImage
{
   friend class  DFALGroup;
   friend class  DFALDataSet;
   friend class  DFALAttribute;
public:
      char* strFileName;
      char* strFileFormat;
      DFAL_UINT64 u64FileSize;
      DFALAccess accessMode;
      DFALLibraryType libraryType;
protected:
      DFALImage();
public:
      virtual void DeleteThis();
public:
      virtual bool InitDFALImage();
      virtual void CreateDFALImage();
public:
      virtual DFALAttribute* GetMetadataX(char* strDomain);
protected:
     virtual DFALAttribute* GetMetadataXMD(char* strDomain);
public:
      virtual DFALDataSet* GetDataSet();
      virtual DFALDataSet* GetDataSet(char* strDataSet);
      virtual DFALGroup* GetRootGroup();
      virtual DFALGroup* GetGroup(char* GroupIndex);
      virtual DFALErr SetMetadata(char* pszMetadataString, char* pszDomain);
      virtual DFALDataSet* CreateDataSet( int nXSize, int nYSize, int
nBands,DFALDataType datatype);
      virtual DFALDataSet* CreateDataSet( int nDims, int* Dims, char*
strDataSet,DFALDataType datatype);
      virtual DFALAttribute* CreateMetadata( int nDims, int* Dims, char*
strDataSet,DFALDataType datatype);
```

```
    virtual DFALErr CreateMetadata(char* strDataSet,DFALDataType datatype,int
buf_size,void* buffer);
    virtual DFALGroup* CreateGroup(char* strGroup);
protected:
    virtual void close();
};
```

4.4. DFALGroup class interface

```
class  DFALGroup
{
    friend class  DFALDataSet;
    friend class  DFALAttribute;
public:
    DFAL_UINT32 u32DataSetCount;
    DFAL_UINT32 u32MetadataCount;
    char* strFileName;
protected:
    DFALGroup();
public:
    virtual void DeleteThis();
public:
    virtual DFALAttribute* GetMetadataX(char* strDomain);
protected:
   virtual DFALAttribute* GetMetadataXMD(char* strDomain);
public:
    virtual DFALDataSet* GetDataSet(char* strDataSet);
   virtual DFALErr SetMetadata(char* pszMetadataString, char* pszDomain);
    virtual DFALDataSet* CreateDataSet( int nDims, int* Dims, char*
strDataSet,DFALDataType datatype);
    virtual DFALAttribute* CreateMetadata( int nDims, int* Dims, char*
strDataSet,DFALDataType datatype);
    virtual DFALErr CreateMetadata(char* strDataSet,DFALDataType datatype,int
buf_size,void* buffer);
    virtual DFALGroup* CreateGroup(char* strGroup);
protected:
    virtual void close();
```

```
};
```

4.5. DFALDataSet class interface

```
class  DFALDataSet
{
    friend class  DFALBand;
    friend class DFALAttribute;
public:
    DFAL_UINT32 u32RasterBandCount;
    DFAL_UINT32 u32RasterXSize;
    DFAL_UINT32 u32RasterYSize;
    DFAL_UINT32 u32RasterZSize;
    DFAL_UINT32 u32RasterDims;
    DFAL_UINT32 u32MetadataCount;
    char* strFileName;
    DFALDataType DataType;
protected:
    DFALDataSet();
public:
    virtual void DeleteThis();
public:
    virtual DFAL_UINT32 GetLongitudeSize();
    virtual void ReadLongitude(DFAL_FLOAT64* buffer);
    virtual DFAL_UINT32 GetLatitudeSize();
    virtual void ReadLatitude(DFAL_FLOAT64* buffer);
    virtual DFAL_UINT32 GetSolarZenithSize();
    virtual void ReadSolarZenith (DFAL_FLOAT64* buffer);
    virtual DFAL_UINT32 GetSatelliteZenithSize();
    virtual void ReadSatelliteZenith (DFAL_FLOAT64* buffer);
    virtual DFAL_UINT32 GetRelativeDirectionZenithSize();
    virtual void ReadRelativeDirectionZenith(DFAL_FLOAT64* buffer);
    virtual DFAL_UINT32 GetVisibleLightCalibrationCoefficientSize();
    virtual void ReadVisibleLightCalibrationCoefficient(DFAL_FLOAT64* buffer);
    virtual DFAL_UINT32 GetInfraredCalibrationCoefficientSize();
    virtual void ReadInfraredCalibrationCoefficient(DFAL_FLOAT64* buffer);
```

```
public:
    virtual char** GetMetadata(char* strDomain);
    virtual DFALAttribute* GetMetadataX(char* strDomain);
protected:
   virtual DFALAttribute* GetMetadataXMD(char* strDomain);
public:
    virtual DFALAttribute* CreateMetadata( int nDims, int* Dims, char*
strDataSet,DFALDataType datatype);
    virtual DFALErr CreateMetadata(char* strDataSet,DFALDataType datatype,int
buf_size,void* buffer);
    virtual const char* GetProjection();
    virtual DFALErr GetGeoTransform(double* dd);
    virtual DFALErr SetMetadata(char** pszMetadataString, char* pszDomain);
    virtual DFALErr SetProjection(char* dd);   virtual DFALErr
SetGeoTransform(double* dd);
    virtual DFALBand* GetRasterBand(int bandIndex);
    virtual DFALErr AddBand(DFALDataType dd);
    virtual DFALErr ReadRaster(int xOff, int yOff, int xSize, int ySize, void* buffer,
DFALDataType buf_type, int bandCount,int *  BandMap);
    virtual DFALErr WriteRaster(int xOff, int yOff, int xSize, int ySize, void* buffer,
DFALDataType buf_type, int bandCount,int *  BandMap);
protected:
    virtual void close();
};
```

4.6. DFALBand class interface

```
class  DFALBand
{
public:
    DFALDataType DataType;
    DFAL_UINT32 u32XSize;
    DFAL_UINT32 u32ySize;
    char* strFileName;
protected:
    DFALBand();
public:
    virtual void DeleteThis();
```

```
public:
    virtual DFALErr ReadRaster(int xOff, int yOff, int xSize, int ySize, void* buffer,
DFALDataType buf_type);
    virtual DFALErr WriteRaster(int xOff, int yOff, int xSize, int ySize, void* buffer,
DFALDataType buf_type);
protected:
    virtual void close();
};
```

4.7. DFALAttribute class interface

```
class  DFALAttribute
{
public:
  // data types
DFALDataType DataType;
 // Width of attribute data
DFAL_UINT32 U32 RasterXSize;
// High Attribute Data
DFAL_UINT32 U32 RasterYSize;
// Third Dimensional Length of Attribute Data
DFAL_UINT32 U32 RasterZSize;
// Dimension of attribute data
DFAL_UINT32 U32 RasterDims;
// Whether the attribute data is a string or not
   DFAL_BOOL isString;
protected:
    DFALAttribute();
public:
  //Clean oneself up
Virtual void DeleteThis ();
Public:
//Read attribute data
Virtual DFALErr ReadRaster (void* buffer);
// Write attribute data
Virtual DFALErr WriteRaster (void* buffer);
Protected:
```

```
Virtual void close ();
};
```

5. Examples of reading and writing HDF5 files

According to the three levels of HDF5 files determined by the specification, examples of reading and writing HDF files are given.
Note: HDF5 writing does not support creating image datasets less than 100*100. Compression block is set to 100*100. If it is less than 100*100, the creation of data sets will fail.

5.1. Examples of reading data in HDF5

```
int _tmain(int argc, _TCHAR* argv[])
{
     char* strFileName = "/mnt/hgfs/VM_Share/linuxhdf/linuxtest_example.hdf";
    // Dimensional information of attributes
   Int A_dims [1];
     // Dimension information of datasetss
     Int D_dims [2];
     Herr_t status;
   // File mode, read-only open
   DFALAccess Access Mode = DFAL_AC_ReadOnly;
   // Use basic library types
   DFAL LibraryType libraryType = DFAL_LT_HDF5;
   // Open the image. The fourth parameter is only useful when using the GDAL library, and you can write a string casually when using the HDF5 library.
    DFALImage* pImage =  OpenImageFile(strFileName, accessMode, libraryType);
   char* strMeta_I_C= "RawDataName";
   // DFAL_DT_CHAR
      DFALAttribute* pAttri_I = pImage->GetMetadataX(strMeta_I_C);
      char* strMeta_I_V = new char[pAttri_I->u32RasterXSize + 1];
      pAttri_I->ReadRaster(strMeta_I_V);
      cout<<strMeta_I_V<<endl;
   // Get group
    char* strGroup = "GeometricCorrection";
      DFALGroup* pGroup = pImage->GetGroup(strGroup);
      // Get the attributes of the group
      char* strMeta_G_C = "ProductTime";
      DFALAttribute* pAttriM_G = pGroup->GetMetadataX(strMeta_G_C);
      char* strMeta_G_V = new char[pAttriM_G->u32RasterXSize + 1];
      pAttriM_G->ReadRaster(strMeta_G_V);
   cout<<strMeta_G_V<<endl;
      // Get  the datasets of the group
      char* strDataSet_G_C = "DataSet_30_1";
      DFALDataSet* pDataSetG = pGroup->GetDataSet(strDataSet_G_C);
```

```
   D_dims[0] = pDataSetG->u32RasterXSize;
      D_dims[1] = pDataSetG->u32RasterYSize;
      DFAL_UINT8* buffer = new DFAL_UINT8[D_dims[0]* D_dims[1]];
      pDataSetG->ReadRaster(0,0, D_dims[0] , D_dims[1] ,buffer,pDataSetG->DataType,0,NULL);
   // Get the attributes of the datasets
      char* strMeta_D_C = "CoversionFactor";
      DFALAttribute* pAttri_D = pDataSetG->GetMetadataX(strMeta_D_C);
   char* strMeta_D_V = new char[pAttri_D->u32RasterXSize + 1];
      pAttri_D->ReadRaster(strMeta_D_V);
   cout<<strMeta_D_V<<" "<<pAttri_D->u32RasterXSize<<endl;
   pAttri_D->DeleteThis();
   pDataSetG->DeleteThis();
   pAttriM_G->DeleteThis();
   pGroup->DeleteThis();
   pAttri_I->DeleteThis();
   pImage->DeleteThis();
}
```

5.2. Examples of writing data in HDF5

```
int _tmain(int argc, _TCHAR* argv[])
{
      char* strFileName = "/mnt/hgfs/VM_Share/linuxhdf/linuxtest_example.hdf";
      // Dimensional information of attributes
     Int A_dims [1];
     / / Dimension information of datasetss
      int D_dims[2];
      herr_t status;
      // File mode, create and delete the original
      DFALAccess accessMode = DFAL_AC_TRUNC;
      // Use basic library types
      DFALLibraryType libraryType = DFAL_LT_HDF5;
   // Create an image, the fourth parameter is only useful when using the GDAL library, and you can write a string casually when using the HDF5 library.
      DFALImage* pImage = CreateImageFile(strFileName, accessMode, libraryType,"HDF");
   char* strMeta_I_C= "RawDataName";
   char* strMeta_I_V = "HJ1A-CCD1-454-80-20100530-L20000314472";
   A_dims[0] = strlen(strMeta_I_V);
   cout<<A_dims[0]<<endl;
   // Create image attributes, data type is DFAL_DT_CHAR
      DFALAttribute* pAttri_I = pImage->CreateMetadata(1,A_dims,strMeta_I_C,DFAL_DT_CHAR);
      pAttri_I->WriteRaster(strMeta_I_V);
     char* strGroup = "GeometricCorrection";
      DFALGroup* pGroup = pImage->CreateGroup(strGroup);
      // Create group attributes
      char* strMeta_G_C = "ProductTime";
      char* strMeta_G_V = "2013175122422";
```

```
        A_dims[0] = strlen(strMeta_G_V);
        DFALAttribute* pAttriM_G = pGroup->CreateMetadata(1,A_dims,strMeta_G_C,DFAL_DT_CHAR);
        pAttriM_G->WriteRaster(strMeta_G_V);
        // Create group datasets
        D_dims[0] = 101;
        D_dims[1] = 101;
        DFAL_UINT8* buffer = new DFAL_UINT8[D_dims[0] * D_dims[1]];
        char* strDataSet_G_C = "DataSet_30_1";
        DFALDataSet* pDataSetG = pGroup->CreateDataSet(2,D_dims,strDataSet_G_C,DFAL_DT_Int8);
        pDataSetG->WriteRaster(0,0, D_dims[0] , D_dims[1] ,buffer,DFAL_DT_Int8,0,NULL);
    // Create attributes for datasets
        char* strMeta_D_C = "CoversionFactor";
        char* strMeta_D_V = "1";
    A_dims[0] = strlen(strMeta_D_V);
        DFALAttribute* pAttri_D = pDataSetG->CreateMetadata(1,A_dims,strMeta_D_C,DFAL_DT_CHAR);
        pAttri_D->WriteRaster(strMeta_D_V);
    pAttri_D->DeleteThis();
    pDataSetG->DeleteThis();
    pAttriM_G->DeleteThis();
    pGroup->DeleteThis();
    pAttri_I->DeleteThis();
    pImage->DeleteThis();
}
```

6. Examples of reading/writing HDF4 files

Note: HDF4 does not have the concept of group. As shown in Figure 9, there are two layers of directories of num_observations_1km datasets (using the concept of HDF5 as nested in two groups) in the image. When reading, the dataset can be obtained directly from the image using the string num_observations_1km. The string after the num_observations_1km semicolon is omitted.

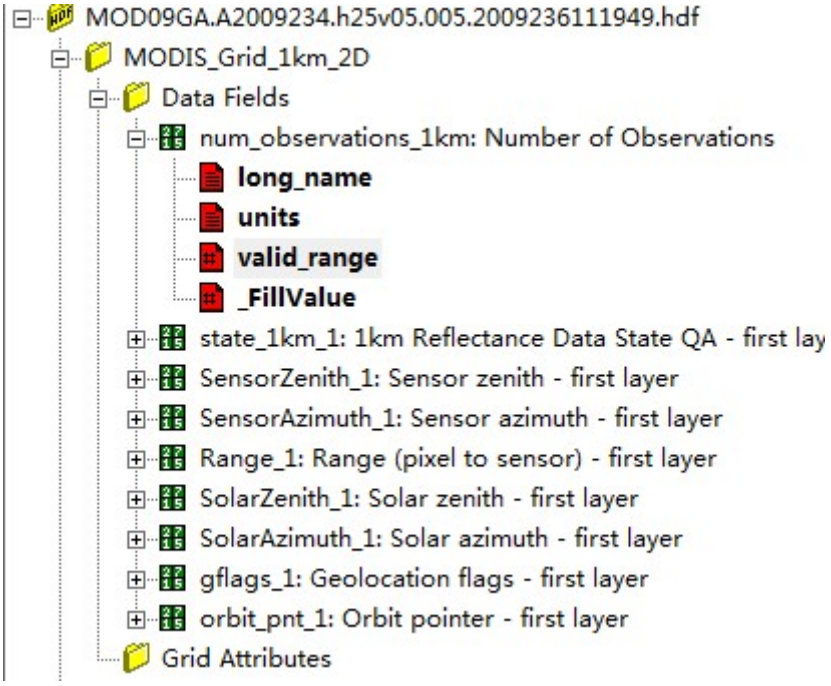


Figure 9 The directories

6.1.Examples of reading data in HDF4

```
int _tmain(int argc, _TCHAR* argv[])
{
    char* strFileName = "/home/lihy64/test_data/MOD09GA.A2009234.h25v05.005.
2009236111949.hdf";
    DFALAccess accessMode = DFAL_AC_ReadOnly;
    DFALLibraryType libraryType = DFAL_LT_HDF4;
    DFALImage* pImage = OpenImageFile(strFileName, accessMode, libraryType);
  //Dataset strings are taken the preceding section of ":"
    DFALDataSet* pDataSet = pImage->GetDataSet("num_observations_1km");
    DFAL_INT16* buffer = new DFAL_INT16[5*5];
    pDataSet->ReadRaster(0,0,5,5,buffer,pDataSet->DataType,1,NULL);
    delete[] buffer;
  // Get the attributes of the datasets
    DFALAttribute *pAttribute_D = pDataSet->GetMetadataX("long_name");
    // Get the attributes of the datasets
    DFALAttribute *pAttribute = pImage->GetMetadataX("StructMetadata.0");
    DFAL_INT8* buffer_A = new DFAL_INT8[pAttribute->u32RasterXSize];
    pAttribute->ReadRaster(buffer_A);
   delete[] buffer_A;
    pAttribute->DeleteThis();
```

```
    pAttribute_D->DeleteThis();
    pDataSet->DeleteThis();
    pImage->DeleteThis();
}
```

6.2. Examples of writing data in HDF4

```
int _tmain(int argc, _TCHAR* argv[])
{
  // preparing data
    int nDims = 2;
    int ndim[2];
    ndim[0] = 5;
    ndim[1] = 5;
    DFAL_INT8* buffer_M = new DFAL_INT8[5*5];
    for (int i=0;i<5*5;i++)
    {
        buffer_M[i] = i;
    }
    DFAL_INT16* buffer = new DFAL_INT16[5*5];
    for (int i=0;i<5;i++)
    {
        for (int j=0;j<5;j++)
        {
            buffer[i*5+j] = i*j;
        }
    }
  // file name
    char* strFileName = "/home/lihy64/test_data/test4.hdf";
  // file mode
    DFALAccess accessMode = DFAL_AC_TRUNC;
    // using basic library types
   DFALLibraryType libraryType = DFAL_LT_HDF4;
    DFALImage* pImage = CreateImageFile(strFileName, accessMode, libraryType,
"HDF");
    pImage->CreateMetadata("Test_M",DFAL_DT_Int8,25,buffer_M);
```

```
    char* hello = "hello hdf4";
    pImage->CreateMetadata("Test_M_C",DFAL_DT_CHAR,11,hello);
    DFALDataSet* pDataSet = pImage->CreateDataSet(nDims,ndim,"test_D",DFAL
_DT_Int16);
    pDataSet->WriteRaster(0,0,5,5,buffer,pDataSet->DataType,1,NULL);
    pDataSet->CreateMetadata("Test_D_M",DFAL_DT_Int8,25,buffer_M);
    pDataSet->DeleteThis();
    pImage->DeleteThis();}
```

7. Examples of reading/writing GDAL files

7.1. Examples of reading data in GDAL

```
int _tmain(int argc, _TCHAR* argv[])
{
    DFAL_INT8* buffer = new DFAL_INT8[5*5];
    char* strFileName = "/home/lihy64/test_data/HJ1A-CCD1-17-64-20120507-
L20000766647-1.TIF";
    DFALAccess accessMode = DFAL_AC_ReadOnly;
   DFALLibraryType libraryType = DFAL_LT_GDAL;
    DFALImage* pImage = OpenImageFile(strFileName, accessMode, libraryType);
    DFALDataSet* pDataSet = pImage->GetDataSet();
    printf("%s\n",pDataSet->GetProjection());
    DFALBand* pBand = pDataSet->GetRasterBand(1);
    pBand->ReadRaster(6550,6550,5,5,buffer,DFAL_DT_Int8);
    delete[] buffer;
    pBand->DeleteThis();
    pDataSet->DeleteThis();
    pImage->DeleteThis();
}
```

7.2. Examples of writing data in GDAL

```
int _tmain(int argc, _TCHAR* argv[])
{
    DFAL_INT8* buffer = new DFAL_INT8[500*500];
    for (int i=0;i<500;i++)
    {
        for (int j=0;j<500;j++)
        {
            buffer[i*500+j] = i*j%255;
        }
    }
    char* strFileName = "/home/lihy64/test_data/linuxgdaltest.tif";
```

```
    char* strFileFormat = "GTiff";
    DFALAccess accessMode = DFAL_AC_TRUNC;
    DFALLibraryType fileFormat = DFAL_LT_GDAL;
    DFALImage* pImage = CreateImageFile(strFileName,
accessMode,fileFormat,strFileFormat);
    DFALDataSet* pDataSet = pImage->CreateDataSet(500,500,3,DFAL_DT_Int8);

    for (int i=0;i<3;i++)
    {
        DFALBand* pBand = pDataSet->GetRasterBand(i+1);
        DFALErr err = pBand->WriteRaster(0,0,500,500,buffer,DFAL_DT_Int8);
        printf("%d\n",err);
        pBand->DeleteThis();
    }
    pDataSet->DeleteThis();
    pImage->DeleteThis();
}
```

8. Examples of reading L1B files

For L1B files, only read is supported.

```
int _tmain(int argc, _TCHAR* argv[])
{
    char* strFileName =
"/home/lihy64/test_data/NA17_AVHRR_HRPT_L1_ORB_MLT_NUL_20090607_02
47_1100M_PJ.L1B";
    DFALAccess accessMode = DFAL_AC_ReadOnly;
    DFALLibraryType libraryType = DFAL_LT_L1B;
    DFALImage* pImage = OpenImageFile(strFileName, accessMode, libraryType);
    DFALDataSet* pDataSet = pImage->GetDataSet();
    DFAL_UINT16* buffer = new DFAL_UINT16[500*500];
    int length = 0;
    // Reading longitude data
   DFAL_FLOAT64* buffer_f = pDataSet->ReadLongitude(length);
    DFALBand* pBand = pDataSet->GetRasterBand(1);
   pBand->ReadRaster(0,0,500,500,buffer,DFAL_DT_UInt16);
    //pBand->DeleteThis();
    pDataSet->DeleteThis();
    pImage->DeleteThis();
}
```